Patient Transport Tracker

Contributors

|  |  |
| --- | --- |
| Luc Zbonack | [lmzbonack@email.arizona.edu](mailto:lmzbonack@email.arizona.edu) |

Executive Summary:

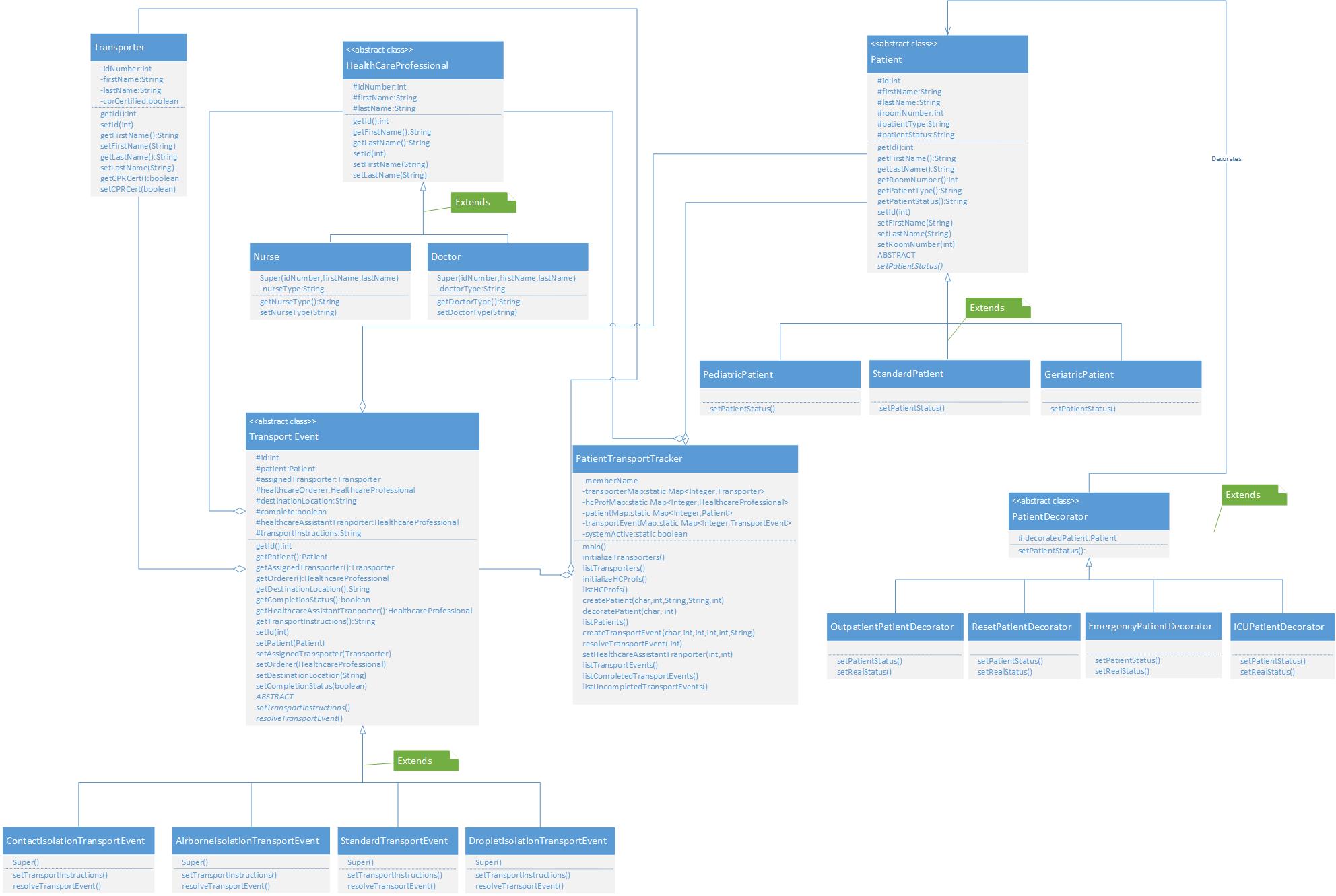
Introduction:

The business scenario for this project was to create the core of an application that can be used to assist transporters in moving patients from their main room in the hospital to a variety of other places for tests/scans etc.

Every hospital has a team of these transporters that are responsible for the movement of patients around the hospital. Transporters will go pick up the patient and move them to the scanning facilities. Then once the scan is complete they will pick the patient back up and take them back to their original room. Sometimes when these transporters must move specific different kinds of patients they are required to received assistance from nurses and/or doctors. It should be noted that because patent statuses change over the course of a hospital stay and transporters need to have the most updated information on patients to do their jobs effectively any system will have to account for potential changing patient information.

Right now, in most hospitals organization of the transporters work is handled by units calling a dispatcher and placing in requests for patient transport. That dispatcher then uses radios to tell specific transporters where to go next. A lightweight computer application could be used to manage this process more effectively increasing the efficiency and speed at which transporters could do their jobs. Enabling better patient outcomes.

System Design:



Patient Transport Tracker contains several different classes and makes use of the strategy design pattern and the decorator design pattern. I will examine the classes and how they fit together to build the system from most simple to most complex. The simplest class is the Transporter class. The purpose of this class is to represent different transporters in the hospital.

Next, we will look at the abstract class HealthCareProfessional. This abstract class is extended by 2 child classes the Nurse class and the Doctor class. The purpose of this abstract class and the classes that extend are the represent the health care professionals that work in the hospital. Currently these professionals are doctors and nurse. However, because it is easy to extend another class from the parent abstract class more types of professionals can be easily added.

The TransportEvent abstract class and its associcated child classes are the first place where the strategy design pattern is utilized. Transport events aggregate instances of the Patient class HealthcareProfessional class and Transporter class to create an object. This object represents the movement of a specific patient within the hospital. This movement is handled by a transporter ordered by a healthcare professional and obviously is for a specific patient. For this reason in the UML diagram the Patient, HealthcareProfessional, and Transporter classes have aggregation relationships with the TransportEvent class.

TransportEvent makes use of the strategy design pattern by defining two abstract methods that its child classes define in different ways. These methods are setTransportInstructions and resolveTransportEvent. This functionality is required because the instructions for different transport events vary and different transport events must be resolved in different ways. For example, some transport events must have specific types of healthcare professionals assist with the transport.

The decorator design pattern is used within this system in tandem with the strategy pattern, as it is typically the case. These two design patterns are used together to create different patient types and statuses. Patient type is set using strategy design pattern and is set first. Then using decorator pattern, different patient statuses can be set After the patient object has already been created. This is the value that comes from using the decorator design pattern it allows the system to keep patient data current and update it quickly which is very important. The Patient abstract class is extended by three different classes that create patients of different types pediatric, standard, and geriatric in this case although more could be added easily.

Then the PatientDecorator class extends the Patient class to decorate it, wrap it in a new class to add data and functionality. This PatientDecorator class is in turn extended by four classes that change the patientStatus attribute within instances of Patient. Currently a patient’s status can be set to ICU, outpatient, or emergency and can be changed or reset to nothing easily. These decorators serve to convey additional information about the patient that can change between different transport events. Keep in mind that multiple transport events are expected during a patients stay within a hospital. These patient statuses are important for transporters to have knowledge of.

Finally, the last class PatientTransportTracker is used to tie the system together and provide a basic command line implementation of the system for users. This class aggregates Patient, TransportEvent, Transporter, and HealthcareProfessional. It will be discussed in more detail in the next section.

System Analysis: